AMENDMENTS TO THE CLAIMS

1. (Previously presented) A method for emulating a plurality of virtual timers in a virtual computer system, the virtual timers being programmable by guest software to generate a plurality of timer events, the method comprising:

receiving programming information from the guest software for programming a first virtual timer;

receiving programming information from the guest software for programming a second virtual timer;

determining when the first virtual timer would generate timer events if it were implemented in a physical computer system, based on the programming information received from the guest software;

determining when the second virtual timer would generate timer events if it were implemented in a physical computer system, based on the programming information received from the guest software; and

generating timer events for the first virtual timer and the second virtual timer in the same combined sequence that they would occur if the first and second virtual timers were implemented in a physical computer system.

2. (Previously presented) The method of claim 1, wherein a catch-up mode is used when the generation of timer events in the virtual computer system is behind the timing of when the timer events would be generated in a physical computer system and a normal mode is used when the generation of timer events in the virtual computer system is caught up to the timing of when the timer events would be generated in a physical computer system; wherein, when the catch-up mode is used, the average rate of timer events in the virtual computer system exceeds the average rate at which timer events would be generated in a physical computer system; and wherein, when the normal mode is used, the average rate of timer events in the virtual computer system is substantially the same as the average rate at which timer events would be generated in a physical computer system.

3. (Previously presented) The method of claim 1, wherein a catch-up mode is used when the generation of timer events in the virtual computer system is behind the timing of when the timer events would be generated in a physical computer system and a normal mode is used when the generation of timer events in the virtual computer system is caught up to the timing of when the timer events would be generated in a physical computer system; wherein, when the catch-up mode is used, the average rate of timer events in the virtual computer system exceeds the average rate at which timer events would be generated in a physical computer system; and wherein, when the normal mode is used, the average rate of timer events in the virtual computer system is substantially the same as the average rate at which timer events would be generated in a physical computer system.

4. (Previously presented) The method of claim 2, wherein, when the normal mode is used, the interval between successive timer events in the virtual computer system is substantially the same as the interval that would occur between the same successive timer events in a physical computer system.

5. (Previously presented) The method of claim 2, wherein the catch-up mode is entered when the generation of timer events in the virtual computer system falls behind the timing of when the timer events would be generated in a physical computer system by a predetermined amount and the normal mode is entered when the generation of timer events in the virtual computer system goes ahead of the timing of when the timer events would be generated in a physical computer system by a predetermined amount.

6. (Currently amended) The method of claim 2, wherein the catch-up mode is entered substantially immediately when the generation of timer events in the virtual computer system falls behind the timing of when the timer events would be generated in a physical computer system and the normal mode is entered substantially immediately

when the generation of timer events in the virtual computer system catches up to the timing of when the timer events would be generated in a physical computer system.

7. (Previously presented) The method of claim 1, wherein, if the guest software attempts to read a count value from a virtual timer, a count value is returned to the guest software that represents a time value that occurs after a time value that is represented by a most recent preceding timer event and before a time value that is represented by a next timer event to occur.

8. (Previously presented) The method of claim 7, wherein the time value that is represented by the count value that is returned to the guest software falls proportionately between the time value that is represented by the most recent preceding timer event and the time value that is represented by the next timer event to occur, based on the proportion at which the time of the attempted reading of the count value falls between the actual time that the most recent preceding timer event was generated and the actual time that the next timer event is scheduled to be generated.

9. (Previously presented) The method of claim 1, wherein the method is performed by keeping track of an apparent time, which represents the time as it would appear to the guest software.

10. (Previously presented) The method of claim 1, wherein the method is performed using a timer event queue.

11. (New) A computer program embodied in a tangible medium, the computer program being executable as part of a virtual computer system, the virtual computer system comprising one or more timer emulators for emulating a plurality of virtual timers, each of the plurality of virtual timers generating one or more timer events, the computer program comprising:

a time coordinator for coordinating the respective timer events of the plurality of virtual timers, the time coordinator:

determining how each of the plurality of virtual timers has been programmed;

based on how each of the virtual timers has been programmed, determining a relative sequence of timer events that would be generated by the virtual timers if they were implemented in a physical computer system; and

notifying the one or more timer emulators when each of the plurality of virtual timers should generate a timer event, so that the timer events are generated in the same combined sequence as if the virtual timers had been implemented in a physical computer system.

12. (New) The computer program of claim 11, wherein the time coordinator uses an apparent time, representing the time as it appears to a software entity that is using the plurality of virtual timers, in determining the relative sequence of timer events that would be generated by the virtual timers if they were implemented in a physical computer system.

13. (New) The computer program of claim 11, wherein the time coordinator uses a timer event queue to determine the relative sequence of timer events that would be generated by the virtual timers if they were implemented in a physical computer system.

14. (New) The computer program of claim 13, wherein the timer event queue maintains a single time value for each of the plurality of virtual timers, representing a time at which the respective virtual timer should generate its next timer event.

15. (New) The computer program of claim 14, wherein the timer event queue comprises a linked list.

16. (New) The computer program of claim 11, wherein the time coordinator has a catch-up mode that is used when the generation of timer events in the virtual computer system is behind the timing of when the timer events would be generated in a physical computer system and a normal mode that is used when the generation of timer events in the virtual computer system is caught up to the timing of when the timer events would be generated in a physical computer system; wherein, when the time coordinator is in the catch-up mode, the average rate of timer events in the virtual computer system exceeds the average rate at which timer events would be generated in a physical computer system; and wherein, when the time coordinator is in the normal mode, the average rate of timer events in the virtual computer system is substantially the same as the average rate at which timer events would be generated in a physical computer system.

17. (New) The computer program of claim 16, wherein, when the time coordinator is in the catch-up mode, the interval between successive timer events in the virtual computer system is substantially proportional to the interval that would occur between the same successive timer events in a physical computer system.

18. (New) The computer program of claim 16, wherein, when the time coordinator is in the normal mode, the interval between successive timer events in the virtual computer system is substantially the same as the interval that would occur between the same successive timer events in a physical computer system.

19. (New) The computer program of claim 16, wherein the time coordinator enters the catch-up mode when the generation of timer events in the virtual computer system falls behind the timing of when the timer events would be generated in a physical computer system by a predetermined amount and the time coordinator enters the normal mode when the generation of timer events in the virtual computer system goes ahead of the timing of when the timer events would be generated in a physical computer system by a predetermined amount.

20. (New) The computer program of claim 16, wherein the time coordinator enters the catch-up mode substantially immediately when the generation of timer events in the virtual computer system falls behind the timing of when the timer events would be generated in a physical computer system and the time coordinator enters the normal mode substantially immediately when the generation of timer events in the virtual computer system catches up to the timing of when the timer events would be generated in a physical computer system.

21. (New) The computer program of claim 11, wherein, if a software entity attempts to read a count value from a virtual timer, the time coordinator provides a value to one of the timer emulators, which causes the timer emulator to return a count value to the software entity that represents a time value that occurs after a time value that is represented by a most recent preceding timer event and before a time value that is represented by a next timer event to occur.

22. (New) The computer program of claim 21, wherein the time value that is represented by the count value that is returned to the software entity falls proportionately between the time value that is represented by the most recent preceding timer event and the time value that is represented by the next timer event to occur, based on the proportion at which the time of the attempted reading of the count value falls between the actual time that the most recent preceding timer event was generated and the actual time that the next timer event is scheduled to be generated.

23. (New) A method for coordinating a plurality of virtual timers in a virtual computer system, the virtual computer system operating within a physical computer system, the method comprising:

receiving programming information for each of the virtual timers, indicating when each of the virtual timers is to generate timer events;

determining when each of the virtual timers would generate timer events if the virtual timers were implemented in a physical computer system; and

causing the virtual timers to generate timer events in the same combined sequence as if the virtual timers had been implemented in a physical computer system.

24. (New) The method of claim 23, further comprising:

using a physical timer in the physical computer system to determine a real time reference that progresses in accordance with the timing of the physical timer;

determining an apparent time that would appear to exist within the virtual computer system based on timing information provided by the virtual timers;

when the apparent time is substantially the same as the real time, generating timer events at substantially the same real time as they would be generated if the virtual timers had been implemented in a physical computer system; and

when the apparent time is substantially behind the real time, generating timer events at a faster rate than they would be generated if the virtual timers had been implemented in a physical computer system, until the apparent time catches up to the real time.

25. (New) The method of claim 24, wherein, when the apparent time is substantially behind the real time, the interval between successive timer events in the virtual computer system is substantially proportional to the interval that would occur between the same successive timer events in a physical computer system.

26. (New) The method of claim 24, wherein timer events are generated at a faster rate than they would be generated if the virtual timers had been implemented in a physical computer system, when the apparent time falls behind the real time by a predetermined amount; and wherein timer events are generated at substantially the same real time as they would be generated if the virtual timers had been implemented

in a physical computer system, when the apparent time goes ahead of the real time by a predetermined amount.

27. (New) The method of claim 24, wherein timer events are generated at a faster rate than they would be generated if the virtual timers had been implemented in a physical computer system substantially immediately when the apparent time falls behind the real time, and wherein timer events are generated at substantially the same real time as they would be generated if the virtual timers had been implemented in a physical computer system substantially immediately when the apparent time catches up to the real time.

28. (New) The method of claim 23, wherein, if a software entity within the virtual computer system attempts to read a count value from a virtual timer, a count value is returned to the software entity that represents a time value that occurs after a time value that is represented by a most recent preceding timer event and before a time value that is represented by a next timer event to occur.

29. (New) The method of claim 28, wherein the time value that is represented by the count value that is returned to the software entity falls proportionately between the time value that is represented by the most recent preceding timer event and the time value that is represented by the next timer event to occur, based on the proportion at which the time of the attempted reading of the count value falls between the actual time that the most recent preceding timer event was generated and the actual time that the next timer event is scheduled to be generated.